Contents

System Architecture

Frontend Layer (Client-Side)

- **Technology:** React with Axios
- **Responsibility:** The React frontend is responsible for rendering the UI and handling user interactions. Axios, a promise-based HTTP client, is used for making HTTP requests to the backend server. Axios simplifies handling request and response interceptors, transforming request and response data, and managing session timeouts.
- **Communication:** Axios sends asynchronous HTTP requests to the backend for data operations. For file operations (upload/download), it may interact with Firebase Storage directly if necessary, although this is often handled through native file API and Firebase SDKs.

Backend Layer (Server-Side)

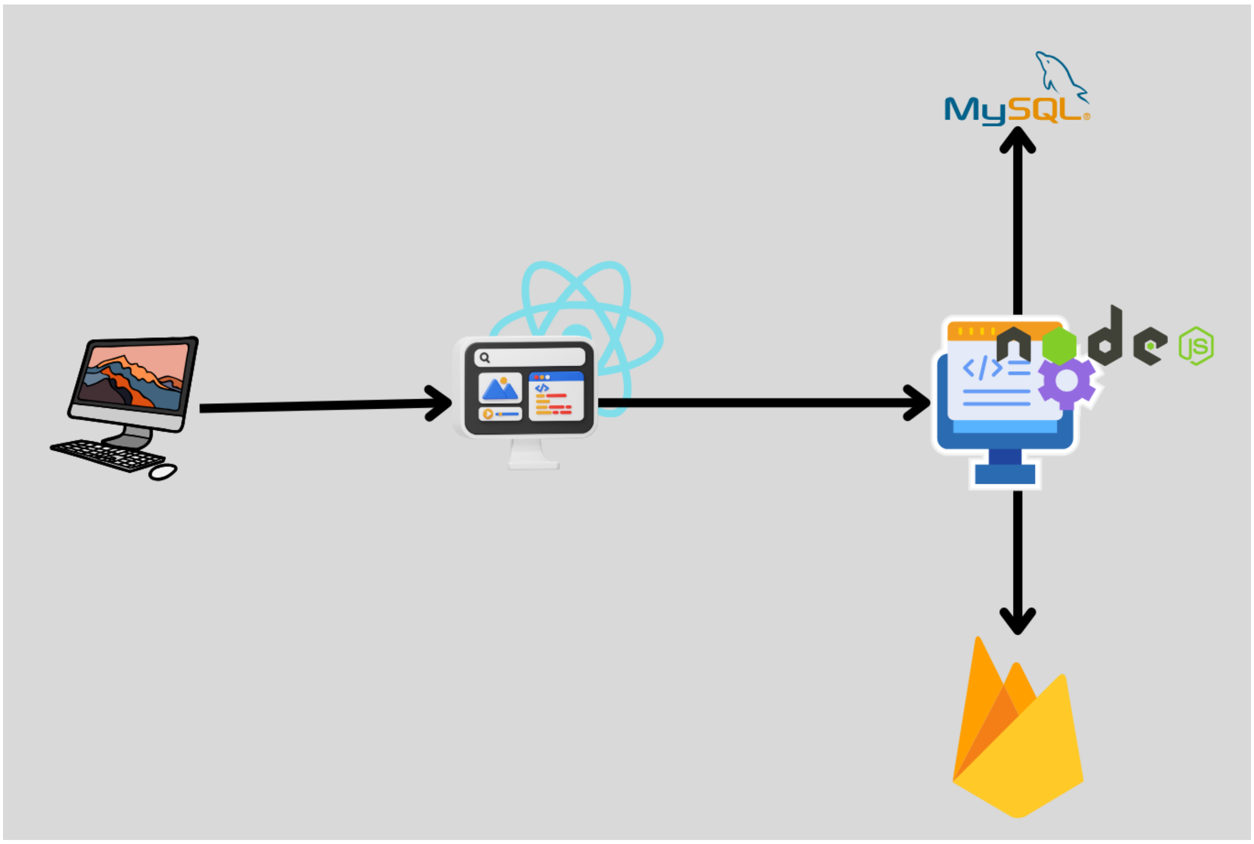
- **Technology:** Node.js with Sequelize ORM
- **Responsibility:** The Node.js backend implements application logic, processes client requests, and manages database operations. Sequelize ORM is used to abstract the database layer, allowing developers to write database queries using JavaScript instead of SQL, which can help prevent SQL injection attacks and improve code maintainability.
- **Communication:** The backend receives API calls from the frontend (via Axios), processes these requests (e.g., user authentication, data processing, validation), and uses Sequelize to perform database operations. The use of Sequelize facilitates connections to MySQL hosted on AWS RDS through a secure and manageable interface, leveraging RDS's scalability and security features.

System Design

- **User Interface (React + Axios):** Provides a dynamic and interactive user experience, using Axios for efficient and flexible communication with the backend.
- **Application Server (Node.js + Sequelize):** Manages core application logic, authentication, and data processing. Sequelize ORM streamlines database interactions with MySQL on AWS RDS, supporting complex queries, transactions, and data mapping with improved security and developer productivity.

- **Database (MySQL on AWS RDS):** Remains the central repository for storing application data. Sequelize enhances interaction with the database by providing a high-level API for CRUD operations, migrations, and more.
- **File Storage (Firebase Storage):** Direct interaction from the frontend for file uploads and downloads, reducing latency and server load. Backend may interact for file management tasks or to generate secure access URLs using Firebase Admin SDK if needed.

This detailed architecture aims to utilize the strengths of each technology (React, Axios, Node.js, Sequelize, MySQL on AWS RDS, and Firebase Storage) to create a scalable, secure, and maintainable web application. Ensuring proper implementation of security best practices, such as secure HTTPS communications, proper authentication and authorization checks, validation and sanitization of inputs, and secure handling of files, is paramount to protecting the application and its users.



API DOCUMENTATION

Authentication

Register User

This API creates new user in the system

URL: POST - BASE_URL/auth/register

SAMPLE-DATA:

```
{
   "userName": "testusername",
   "imageURL": "test url",
   "email": "test@test.com",
   "phoneNumber": "testPhonenumber",
   "password": "test password",
   "address": "test address"
}
```

SAMPLE-REPONSE:

```
{
   "status": 200,
   "message": "user registered successfully!!!"
}
```

Login

This API lets user login to the system

URL: POST - BASE_URL/auth/login

SAMPLE-DATA:

```
{
   "email": "test@test.com",
   "password": "test password"
}
```

SAMPLE-RESPONSE:

```
{
   "message": "Login successful!",
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoiNmVkMDdmNTctNzY5MS00ZDQyLWE1YWEtOGZlNjJlMjVh
NjQxIiwiaWF0IjoxNzEyNDEwMTQyLCJleHAiOjE3MTI0MTE5NDJ9.FrbYlZXr9J-
NSvkMdEX_khFG_rVRhsS1sQowAijlfaY",
   "refreshToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoiNmVkMDdmNTctNzY5MS00ZDQyLWE1YWEtOGZlNjJlMjVh
NjQxIiwiaWF0IjoxNzEyNDEwMTQyLCJleHAiOjE3MTI0OTY1NDJ9.oPMt6ewqola-
5gEZFIFWvIS_GX81Vo4PPzqGHvcUJ7s",
   "user": {
      "uuid": "6ed07f57-7691-4d42-a5aa-8fe62e25a641",
      "userName": "testusername",
      "imageURL": "test url",
      "email": "test2@test.com",
      "phoneNumber": "testPhonenumber",
      "address": "test address",
      "isAdmin": true,
      "isSuperAdmin": false,
      "refreshToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoiNmVkMDdmNTctNzY5MS00ZDQyLWE1YWEtOGZlNjJlMjVh
NjQxIiwiaWF0IjoxNzEyNDEwMTQyLCJleHAiOjE3MTI0OTY1NDJ9.oPMt6ewqola-
5gEZFIFWvIS_GX81Vo4PPzqGHvcUJ7s"
   }
}
```

Logout

This API logs out the user from the system

URL: POST - BASE_URL/auth/logout

Job

Add Job

This api adds a new job posting to the system

URL: POST - BASE_URL/api/jobs

SAMPLE-DATA:

```
{
   "institutionName": "testname2",
   "imageURL": "",
   "post": "testpost",
   "level": "",
   "vacancy": 2,
   "type": "testtype",
   "salary": 1222,
   "deadline": "2024-10-11",
   "state": "OH",
   "city": "test city",
   "venue": "test venue",
   "education": "test education",
   "experience": "test experience",
   "specifications": "test specifications, fwejhrf,fewf",
   "description": "test description, ejfhef,"
}
```

SAMPLE-RESPONSE:

```
{
   "message": "Job created successfully",
   "job": {
      "uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
      "institutionName": "testname2",
      "imageURL": "",
      "post": "testpost",
      "level": "",
      "vacancy": 2,
      "type": "testtype",
      "salary": 1222,
      "deadline": "2024-10-11T00:00:00.000Z",
      "state": "OH",
      "city": "test city",
      "venue": "test venue",
      "education": "test education",
      "experience": "test experience",
      "specifications": "test specifications, fwejhrf,fewf",
        "description": "test description, ejfhef,",
        "updatedAt": "2024-04-09T04:51:53.475Z",
        "createdAt": "2024-04-09T04:51:53.475Z"
    }
}
```

Update job

This API updates a job posting in the system

URL: PUT - BASE_URL /api/jobs/1354ca10-5f99-489e-8148-24ba543e51fe (uuid of job)

SAMPLE-DATA:

```
{
    "institutionName": "testname updated3",
    "post": "testpost updated",
    "level": "testlevel",
    "vacancy": 2,
    "type": "testtype",
    "salary": 1222,
    "deadline": "2024-10-1",
    "state": "OH",
    "city": "test city updated",
    "venue": "test venue",
    "education": "test education",
    "experience": "test experience",
    "specifications": "test specifications",
    "description": "test description"
}
```

SAMPLE-REPONSE:

```
{
    "message": "Job updated successfully",
    "udpatedJob": {
        "uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
        "institutionName": "updated testname2",
        "imageURL": "",
        "post": "testpost",
        "level": "",
        "vacancy": 2,
        "type": "testtype",
        "salary": 1222,
        "deadline": "2024-10-11T00:00:00.000Z",
        "state": "OH",
        "city": "test city",
        "venue": "test venue",
        "education": "test education",
        "experience": "test experience",
        "specifications": "test specifications, fwejhrf,fewf",
        "description": "test description, ejfhef,",
        "createdAt": "2024-04-09T04:51:53.000Z",
        "updatedAt": "2024-04-09T04:53:49.000Z"
    }
}
```

Fetch All Jobs

This api fetches all the job postings in the system

URL: GET - BASE_URL/api/jobs (fetches all jobs)

SAMPLE-REPONSE:

```
[
    {
        "uuid": "bc935f5d-4d2b-40d1-b731-79914bdbb5d2",
        "institutionName": "testname2",
        "imageURL": "",
        "post": "testpost",
        "level": "",
        "vacancy": 2,
        "type": "testtype",
        "salary": 1222,
        "deadline": "2024-10-11T00:00:00.000Z",
        "state": "OH",
        "city": "test city",
        "venue": "test venue",
        "education": "test education",
        "experience": "test experience",
        "specifications": "test specifications",
        "description": "test description",
        "createdAt": "2024-03-23T21:21:33.000Z",
        "updatedAt": "2024-03-23T21:21:33.000Z"
    },
    {
        "uuid": "9407e32d-5032-4bb6-ab71-48a7f8222d09",
        "institutionName": "testname2",
        "imageURL": "",
        "post": "testpost",
        "level": "",
        "vacancy": 2,
        "type": "testtype",
        "salary": 1222,
        "deadline": "2024-10-11T00:00:00.000Z",
        "state": "OH",
        "city": "test city",
        "venue": "test venue",
        "education": "test education",
        "experience": "test experience",
        "specifications": "test specifications, fwejhrf,fewf",
        "description": "test description, ejfhef,",
        "createdAt": "2024-04-09T04:46:11.000Z",
        "updatedAt": "2024-04-09T04:46:11.000Z"
    },
    {
        "uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
        "institutionName": "updated testname2",
        "imageURL": "",
        "post": "testpost",
        "level": "",
        "vacancy": 2,
        "type": "testtype",
        "salary": 1222,
        "deadline": "2024-10-11T00:00:00.000Z",
        "state": "OH",
        "city": "test city",
        "venue": "test venue",
        "education": "test education",
        "experience": "test experience",
        "specifications": "test specifications, fwejhrf,fewf",
        "description": "test description, ejfhef,",
        "createdAt": "2024-04-09T04:51:53.000Z",
        "updatedAt": "2024-04-09T04:53:49.000Z"
    }
]
```

Fetch Job by ID

This api fetches a job by its uuid

URL: GET - BASE_URL/api/jobs/ 0be77d95-a6db-4fb2-8b6c-4bc2228311fd

(uuid of job)

SAMPLE-REPONSE:

```
{
    "uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
    "institutionName": "updated testname2",
    "imageURL": "",
  "post": "testpost",
  "level": "",
  "vacancy": 2,
  "type": "testtype",
  "salary": 1222,
  "deadline": "2024-10-11T00:00:00.000Z",
  "state": "OH",
  "city": "test city",
  "venue": "test venue",
  "education": "test education",
  "experience": "test experience",
  "specifications": "test specifications, fwejhrf,fewf",
  "description": "test description, ejfhef,",
  "createdAt": "2024-04-09T04:51:53.000Z",
  "updatedAt": "2024-04-09T04:53:49.000Z"
}
```

Delete Job by ID

Delete

URL: DELETE - BASE_URL/api/jobs/9c230f4a-93d8-4803-980c-f216247cf268 (uuid of job)

SAMPLE-REPONSE:

```
{
  "message": "Job deleted successfully"
}
```

Accommodation

Add Accommodation

This api adds a new accommodation posting to the system

URL: POST - BASE_URL/api/accommodations

SAMPLE-DATA:

```
{
  "roomType": "test room Type",
  "imageURL": "",
  "moveIn": "2024-10-11",
  "rent": 1111,
  "space": "",
  "amenities": "",
  "rules": "test rules",
    "location": "test location"
}
```

SAMPLE-RESPONSE:

```
{
    "message": "Accommodation created successfully",
    "accommodation": {
        "uuid": "bfab7be5-fd6a-4211-8e65-df2599ee7056",
        "roomType": "test room Type",
        "imageURL": "",
        "moveIn": "2024-10-11T00:00:00.000Z",
        "rent": 1111,
        "space": "",
        "amenities": "",
        "rules": "test rules",
        "location": "test location",
        "updatedAt": "2024-04-09T04:59:15.978Z",
        "createdAt": "2024-04-09T04:59:15.978Z"
    }
}
```

Update Accommodation

This API updates a accommodation posting in the system

URL: PUT - BASE_URL /api/jobs/1354ca10-5f99-489e-8148-24ba543e51fe (uuid of job)

SAMPLE-DATA:

```
{
    "roomType": "test room Type update",
    "imageURL": "test update",
    "moveIn": "2024-1-19",
    "rent": 1111,
    "space": "",
    "amenities": "",
    "rules": "test rules",
    "location": "test location"
}
```

SAMPLE-REPONSE:

```
{
    "message": "Accommodation updated successfully",
    "udpatedAccommodation": {
        "uuid": "bfab7be5-fd6a-4211-8e65-df2599ee7056",
        "roomType": "test room Type update",
        "imageURL": "test update",
        "moveIn": "2024-01-19T05:00:00.000Z",
        "rent": 1111,
        "space": "",
        "amenities": "",
        "rules": "test rules",
        "location": "test location",
        "createdAt": "2024-04-09T04:59:15.000Z",
        "updatedAt": "2024-04-09T05:05:49.000Z"
    }
}
```

Fetch All Accommodations

This api fetches all the accommodation postings in the system

URL: GET - BASE_URL/api/accommodations (fetches all jobs)

SAMPLE-REPONSE:

```
[
    {
        "uuid": "0e4bef60-c10c-4769-ae4c-c78317b7ebd4",
        "roomType": "test room Type",
        "imageURL": "",
        "moveIn": "2024-10-11T00:00:00.000Z",
        "rent": 1111,
        "space": "",
        "amenities": "",
        "rules": "test rules",
        "location": "test location",
        "createdAt": "2024-03-26T23:10:30.000Z",
        "updatedAt": "2024-03-26T23:10:30.000Z"
    },
    {
        "uuid": "bfab7be5-fd6a-4211-8e65-df2599ee7056",
        "roomType": "test room Type update",
        "imageURL": "test update",
        "moveIn": "2024-01-19T05:00:00.000Z",
        "rent": 1111,
        "space": "",
        "amenities": "",
        "rules": "test rules",
        "location": "test location",
        "createdAt": "2024-04-09T04:59:15.000Z",
        "updatedAt": "2024-04-09T05:05:49.000Z"
    }
]
```

Fetch Accommodation by ID

This api fetches a accommodation by its uuid

URL: GET - BASE_URL/api/accommodations/bfab7be5-fd6a-4211-8e65-df2599ee7056

(uuid of accommodation)

SAMPLE-REPONSE:

```
{
    "uuid": "bfab7be5-fd6a-4211-8e65-df2599ee7056",
    "roomType": "test room Type update",
    "imageURL": "test update",
    "moveIn": "2024-01-19T05:00:00.000Z",
    "rent": 1111,
    "space": "",
    "amenities": "",
    "rules": "test rules",
    "location": "test location",
    "createdAt": "2024-04-09T04:59:15.000Z",
    "updatedAt": "2024-04-09T05:05:49.000Z"
}
```

Delete Accommodation by ID

Delete

URL: DELETE - BASE_URL/api/accommodations/9c230f4a-93d8-4803-980c-f216247cf268 (uuid of accommodation)

SAMPLE-REPONSE:

```
{
    "message": "Accommodation deleted successfully"
}
```

Event

Add Event

This api adds a new event posting to the system

URL: POST - BASE_URL/api/events

SAMPLE-DATA:

```
{
   "eventName": "testname",
   "imageURL": "",
   "eventOrganizer": "test organizer",
   "eventTag": "tag1, tag2",
   "date": "2024-10-12",
   "startTime": "22:22",
   "endTime": "22:23",
   "state": "OH",
   "city": "test city",
   "venue": "test venue",
   "description": "test description"
}
```

SAMPLE-RESPONSE:

```
{
   "message": "Event created successfully",
   "event": {
      "uuid": "a0948032-3e7e-4ccc-947f-9071e3efc164",
      "eventName": "testname",
      "imageURL": "",
      "eventOrganizer": "test organizer",
      "eventTag": "tag1, tag2",
      "date": "2024-10-12T00:00:00.000Z",
      "startTime": "22:22",
      "endTime": "22:23",
      "state": "OH",
      "city": "test city",
      "venue": "test venue",
      "description": "test description",
      "updatedAt": "2024-04-09T05:14:22.680Z",
      "createdAt": "2024-04-09T05:14:22.680Z"
   }
}
```

Update Event

This API updates a accommodation posting in the system

URL: PUT - BASE_URL/api/events/1354ca10-5f99-489e-8148-24ba543e51fe (uuid of job)

SAMPLE-DATA:

```
{
   "eventName": "testname updated",
   "imageURL": "updated",
   "eventOrganizer": "test organizer",
   "eventTag": "tag1, tag2",
   "date": "2024-10-12",
   "startTime": "22:22",
   "endTime": "22:25",
   "state": "OH",
   "city": "test city",
   "venue": "test venue",
   "description": "test description"
}
```

SAMPLE-REPONSE:

```
{
   "message": "Event updated successfully",
   "udpatedEvent": {
      "uuid": "a0948032-3e7e-4ccc-947f-9071e3efc164",
      "eventName": "testname updated",
      "imageURL": "updated",
      "eventOrganizer": "test organizer",
      "eventTag": "tag1, tag2",
      "date": "2024-10-12T00:00:00.000Z",
      "startTime": "22:22",
      "endTime": "22:25",
      "state": "OH",
      "city": "test city",
      "venue": "test venue",
      "description": "test description",
      "createdAt": "2024-04-09T05:14:22.000Z",
      "updatedAt": "2024-04-09T05:16:07.000Z"
   }
}
```

Fetch All Events

This api fetches all the events postings in the system

URL: GET - BASE_URL/api/events (fetches all events)

SAMPLE-REPONSE:

```
[
  {
    "uuid": "a0948032-3e7e-4ccc-947f-9071e3efc164",
    "eventName": "testname updated",
    "imageURL": "updated",
    "eventOrganizer": "test organizer",
    "eventTag": "tag1, tag2",
    "date": "2024-10-12T00:00:00.000Z",
    "startTime": "22:22",
    "endTime": "22:25",
    "state": "OH",
    "city": "test city",
    "venue": "test venue",
    "description": "test description",
    "createdAt": "2024-04-09T05:14:22.000Z",
    "updatedAt": "2024-04-09T05:16:07.000Z"
  }
]
```

Fetch Event by ID

This api fetches a accommodation by its uuid

URL: GET - BASE_URL/api/events/bfab7be5-fd6a-4211-8e65-df2599ee7056

(uuid of accommodation)

SAMPLE-REPONSE:

```
{
  "uuid": "a0948032-3e7e-4ccc-947f-9071e3efc164",
  "eventName": "testname updated",
  "imageURL": "updated",
  "eventOrganizer": "test organizer",
  "eventTag": "tag1, tag2",
  "date": "2024-10-12T00:00:00.000Z",
  "startTime": "22:22",
  "endTime": "22:25",
  "state": "OH",
  "city": "test city",
  "venue": "test venue",
  "description": "test description",
  "createdAt": "2024-04-09T05:14:22.000Z",
  "updatedAt": "2024-04-09T05:16:07.000Z"
}
```

Delete Accommodation by ID

Delete

URL: DELETE - BASE_URL/api/events/9c230f4a-93d8-4803-980c-f216247cf268 (uuid of accommodation)

SAMPLE-REPONSE:

```
{
  "message": "Accommodation deleted successfully"
}
```

Job Application

Create a Job Application

This api creates a job application for a user for a job

URL: POST - BASE_URL/api/jobApplications

SAMPLE-DATA:

```
{
  "job_uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
  "fileURL": "fileURL"
}
```

SAMPLE-REPONSE:

```
{
  "message": "JobApplication created successfully",
  "jobApplication": {
    "uuid": "3ea277d6-9928-4e1b-9269-b3361c54ad48",
    "job_uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
    "fileURL": "fileURL",
    "user_uuid": "6530edbb-b9e7-41ef-8aee-cbdf957f1a5e",
    "updatedAt": "2024-04-09T05:28:04.941Z",
    "createdAt": "2024-04-09T05:28:04.941Z"
  }
}
```

Get All Job Application For Job

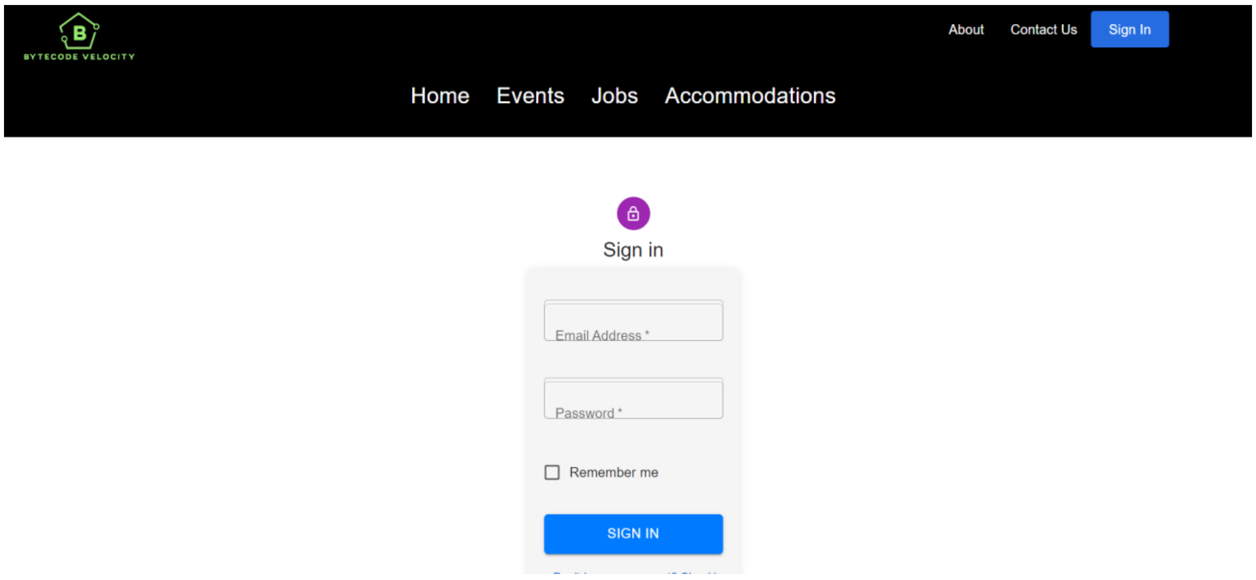
This API gets all the applications for a particular job

URL: GET - BASE_URL/api/jobApplications/job/0be77d95-a6db-4fb2-8b6c-4bc2228311fd

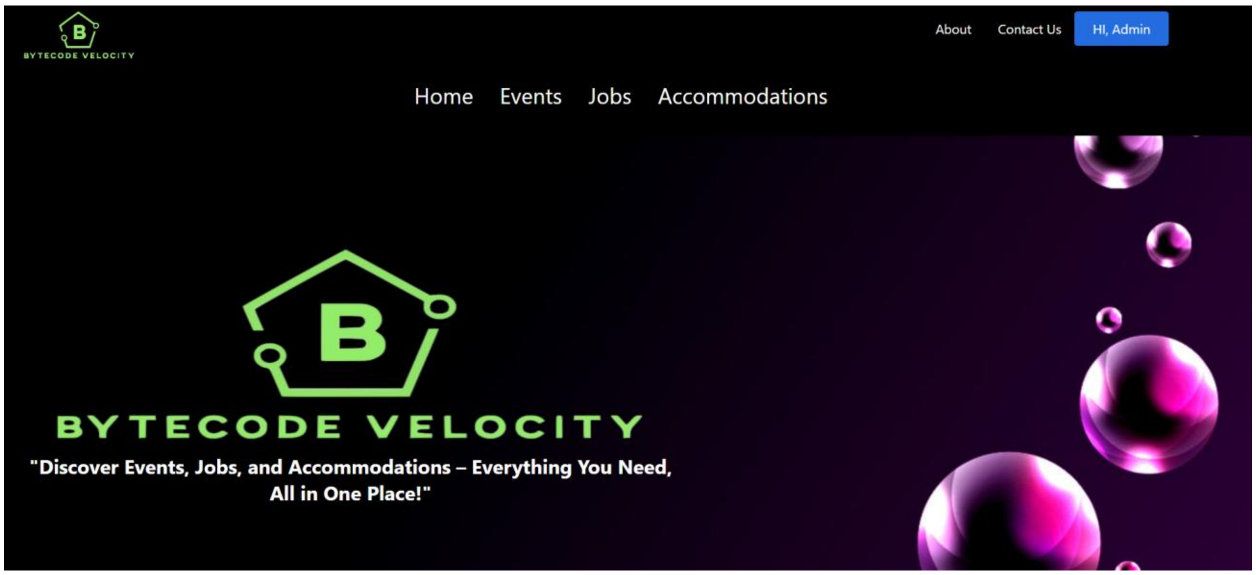
SAMPLE-RESPONSE:

```
[
   {
      "uuid": "3ea277d6-9928-4e1b-9269-b3361c54ad48",
      "job_uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
      "user_uuid": "6530edbb-b9e7-41ef-8aee-cbdf957f1a5e",
      "fileURL": "fileURL",
      "createdAt": "2024-04-09T05:28:04.000Z",
      "updatedAt": "2024-04-09T05:28:04.000Z"
   },
   {
      "uuid": "33458df3-a166-4727-8f44-90c85d7aa23a",
      "job_uuid": "0be77d95-a6db-4fb2-8b6c-4bc2228311fd",
      "user_uuid": "6530edbb-b9e7-41ef-8aee-cbdf957f1a5e",
      "fileURL": "fileURL2",
      "createdAt": "2024-04-09T05:30:11.000Z",
      "updatedAt": "2024-04-09T05:30:11.000Z"
   }
]
```

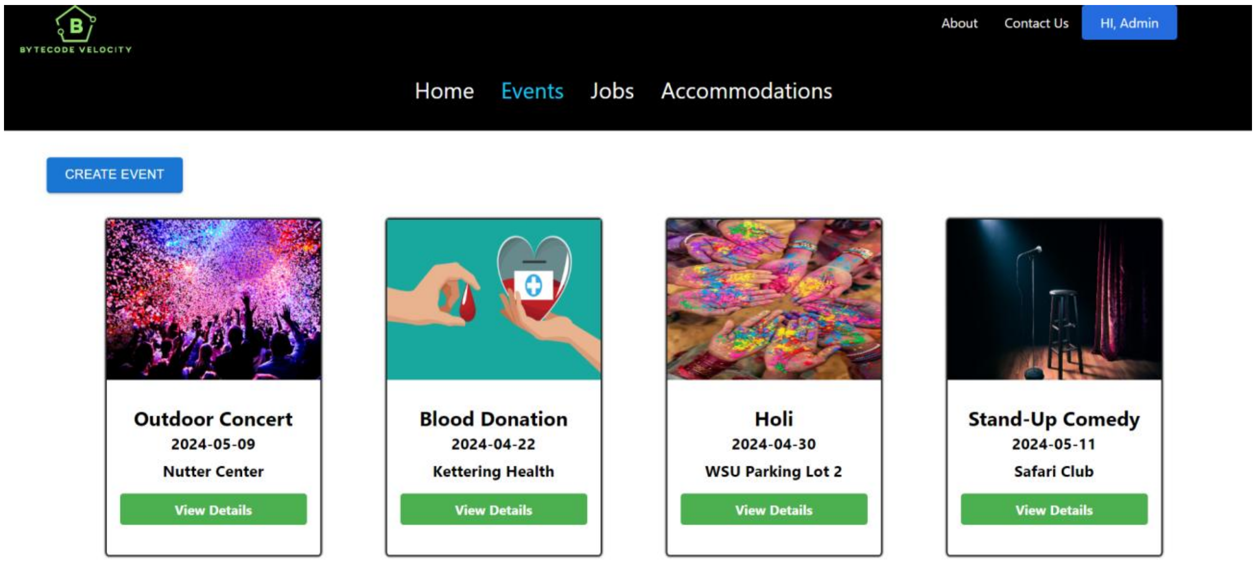
Application View



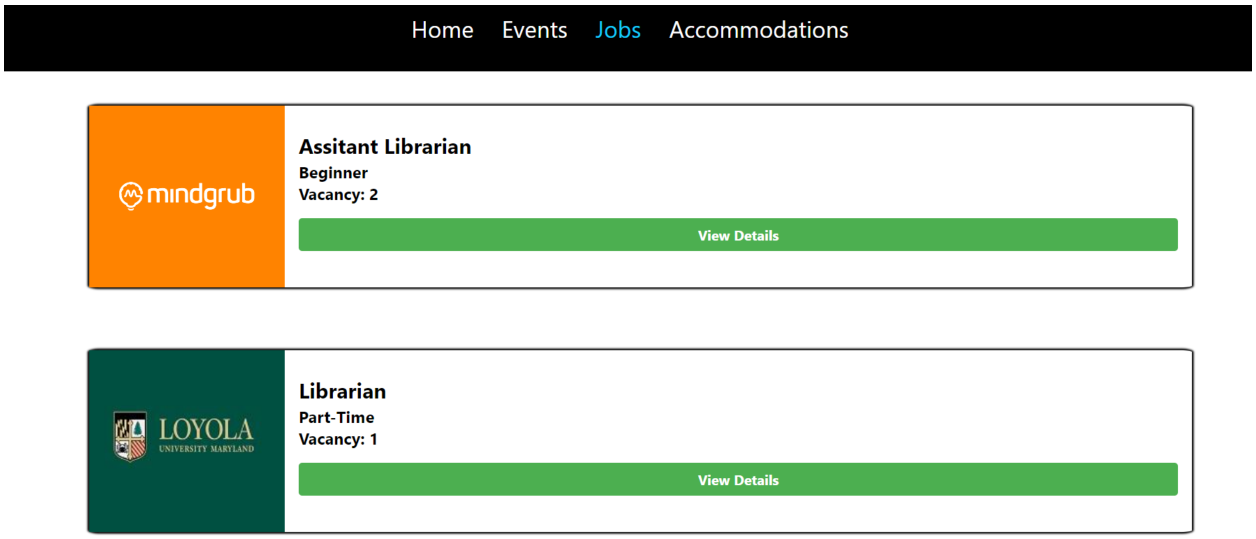
Login page



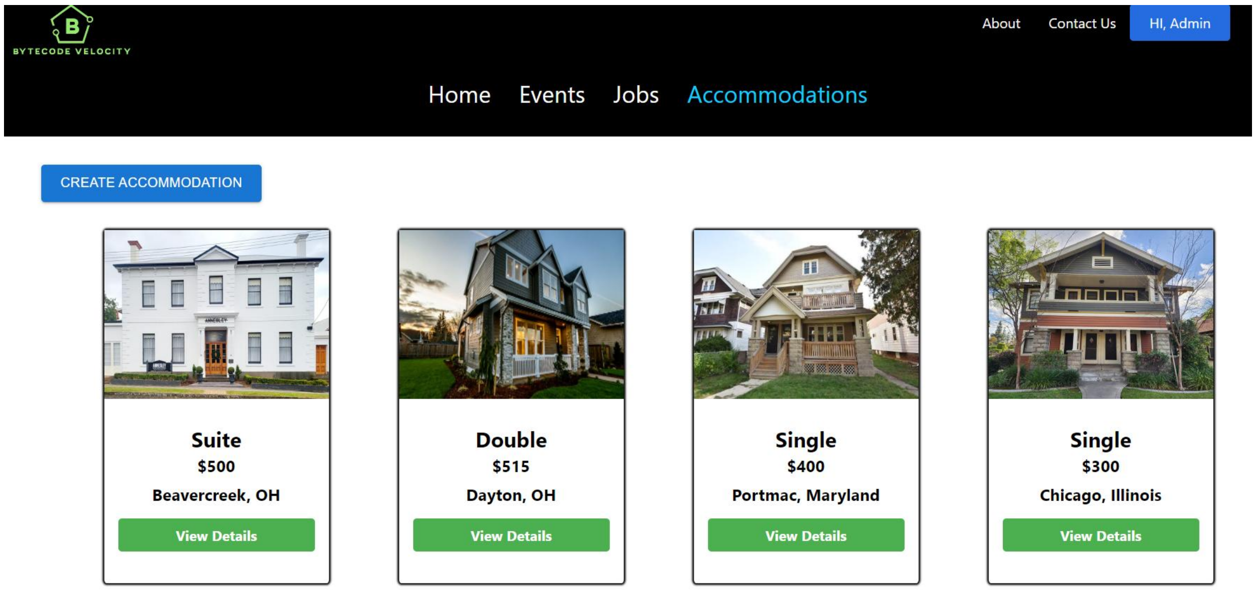
Home page



Events page



Jobs Page



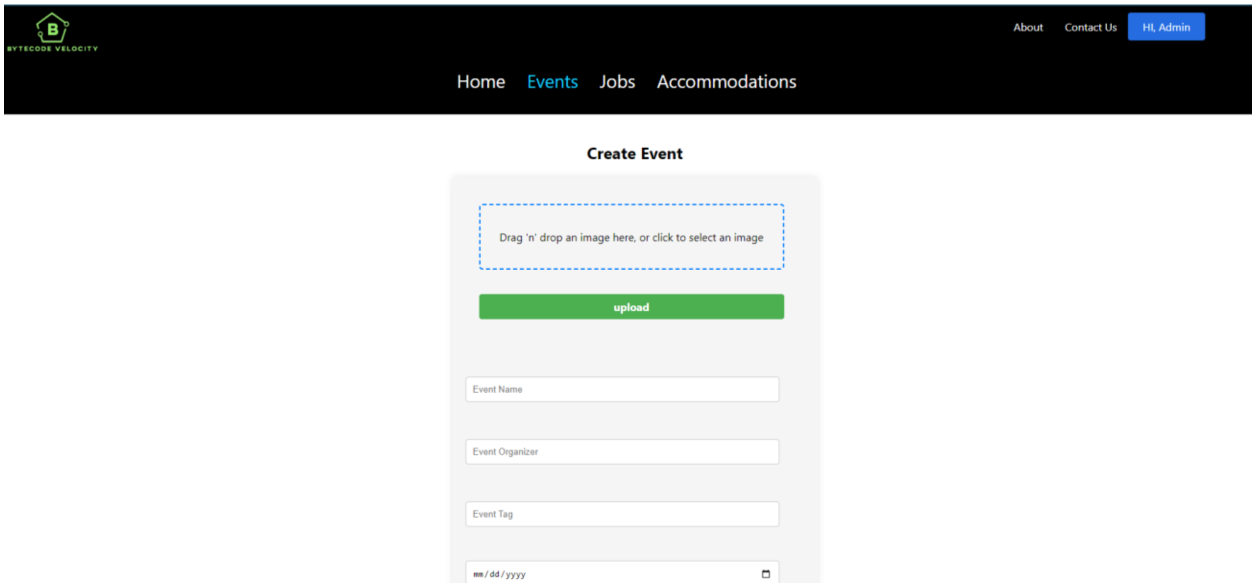
Suite
$500
Beavercreek, OH
View Details

Double
$515
Dayton, OH
View Details

Single
$400
Portmac, Maryland
View Details

Single
$300
Chicago, Illinois
View Details

Accommodation Page



Create Event

Drag 'n' drop an image here, or click to select an image

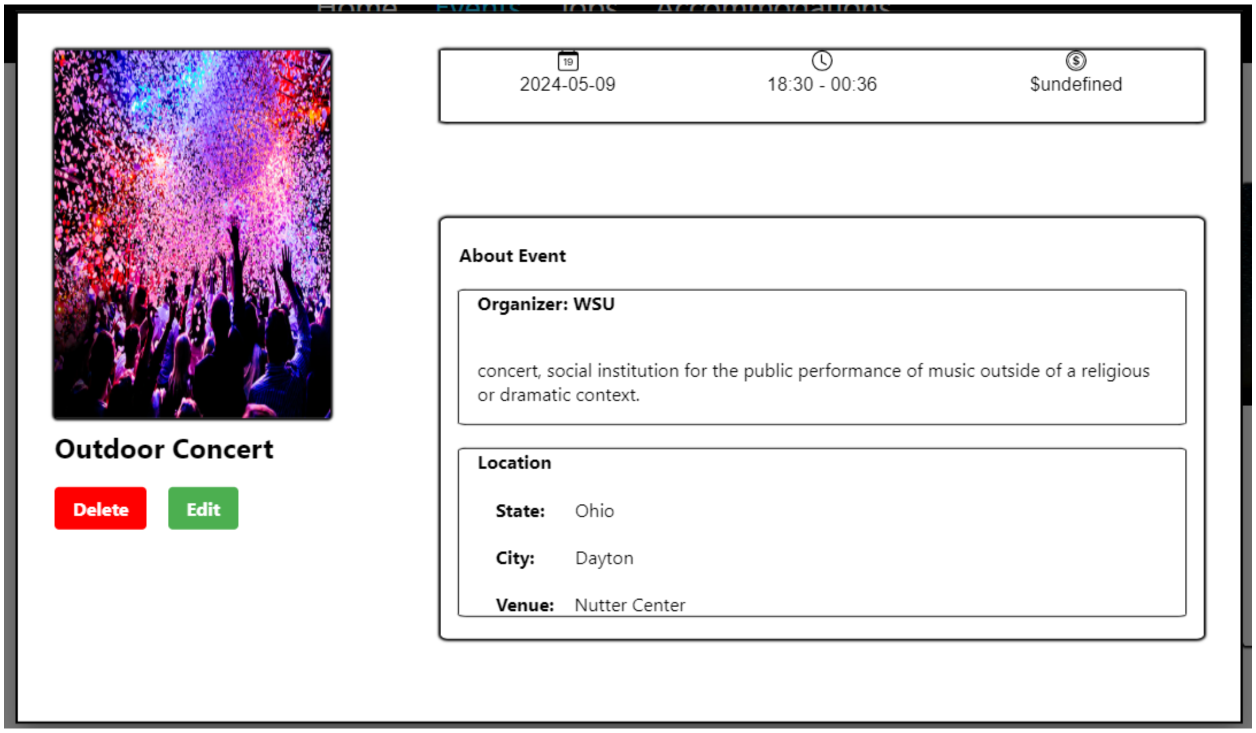
Event Name

Event Organizer

Event Tag

mm / dd / yyyy

Create Event Page



Outdoor Concert

2024-05-09 | 18:30 - 00:36 | $undefined

About Event

Organizer: WSU

concert, social institution for the public performance of music outside of a religious or dramatic context.

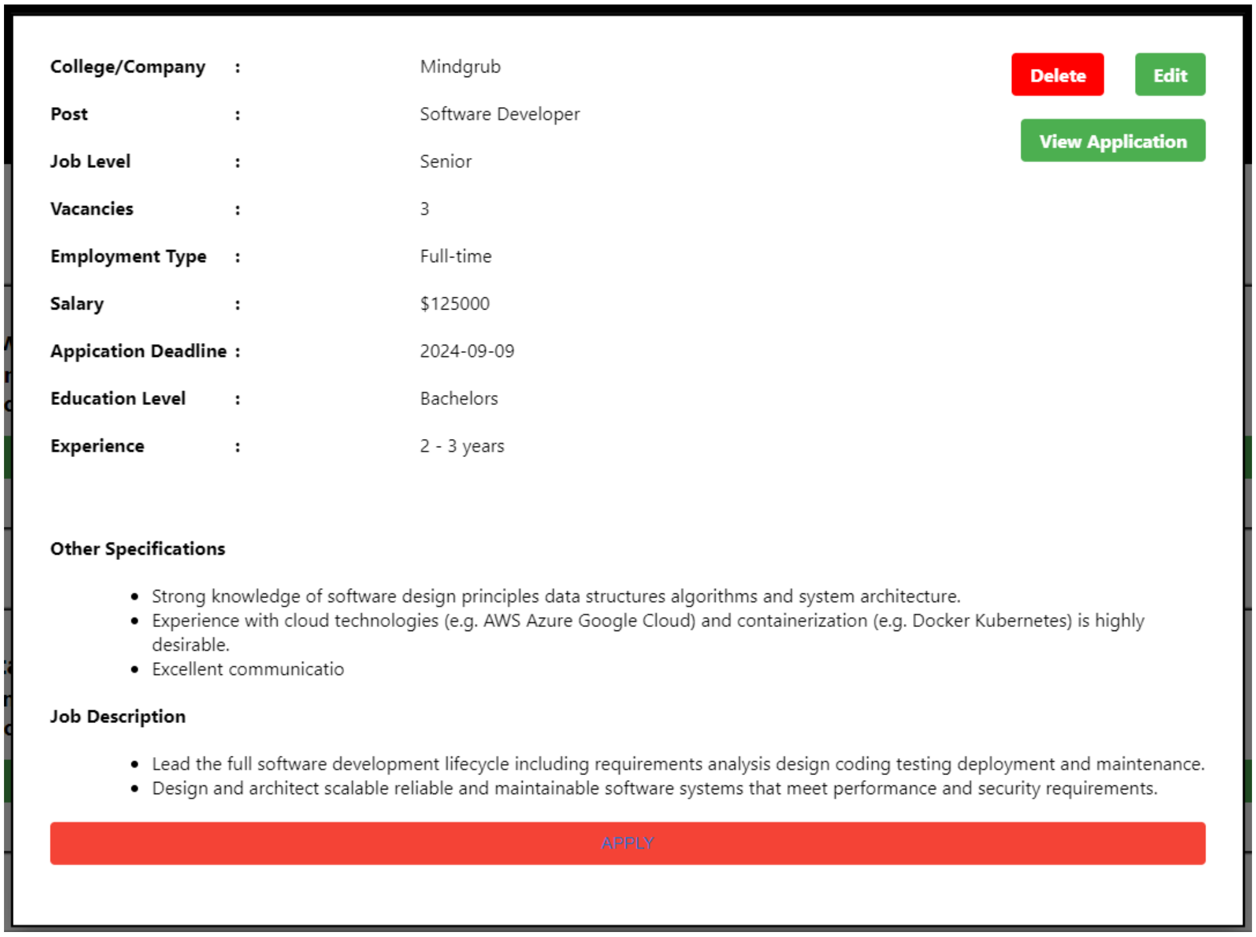
Location

State: Ohio

City: Dayton

Venue: Nutter Center

Event Detail Page



College/Company	:	Mindgrub
Post	:	Software Developer
Job Level	:	Senior
Vacancies	:	3
Employment Type	:	Full-time
Salary	:	$125000
Appication Deadline	:	2024-09-09
Education Level	:	Bachelors
Experience	:	2 - 3 years

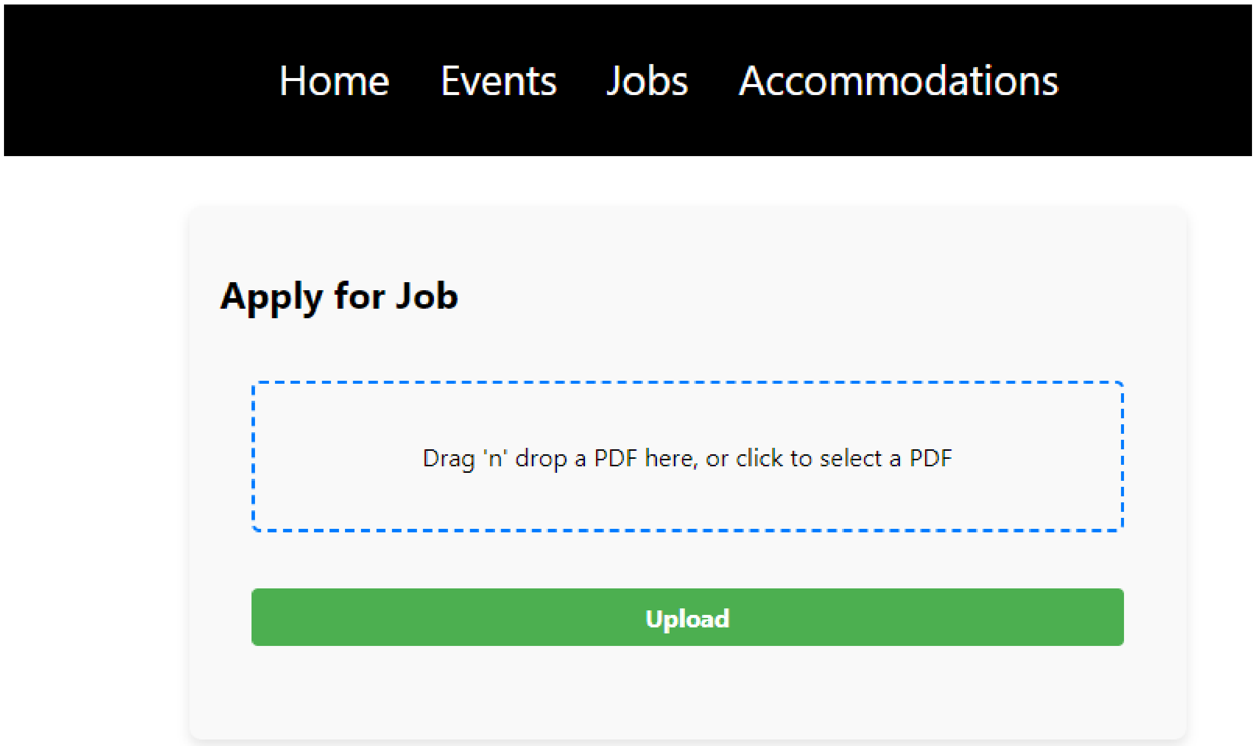
Other Specifications

- Strong knowledge of software design principles data structures algorithms and system architecture.
- Experience with cloud technologies (e.g. AWS Azure Google Cloud) and containerization (e.g. Docker Kubernetes) is highly desirable.
- Excellent communicatio

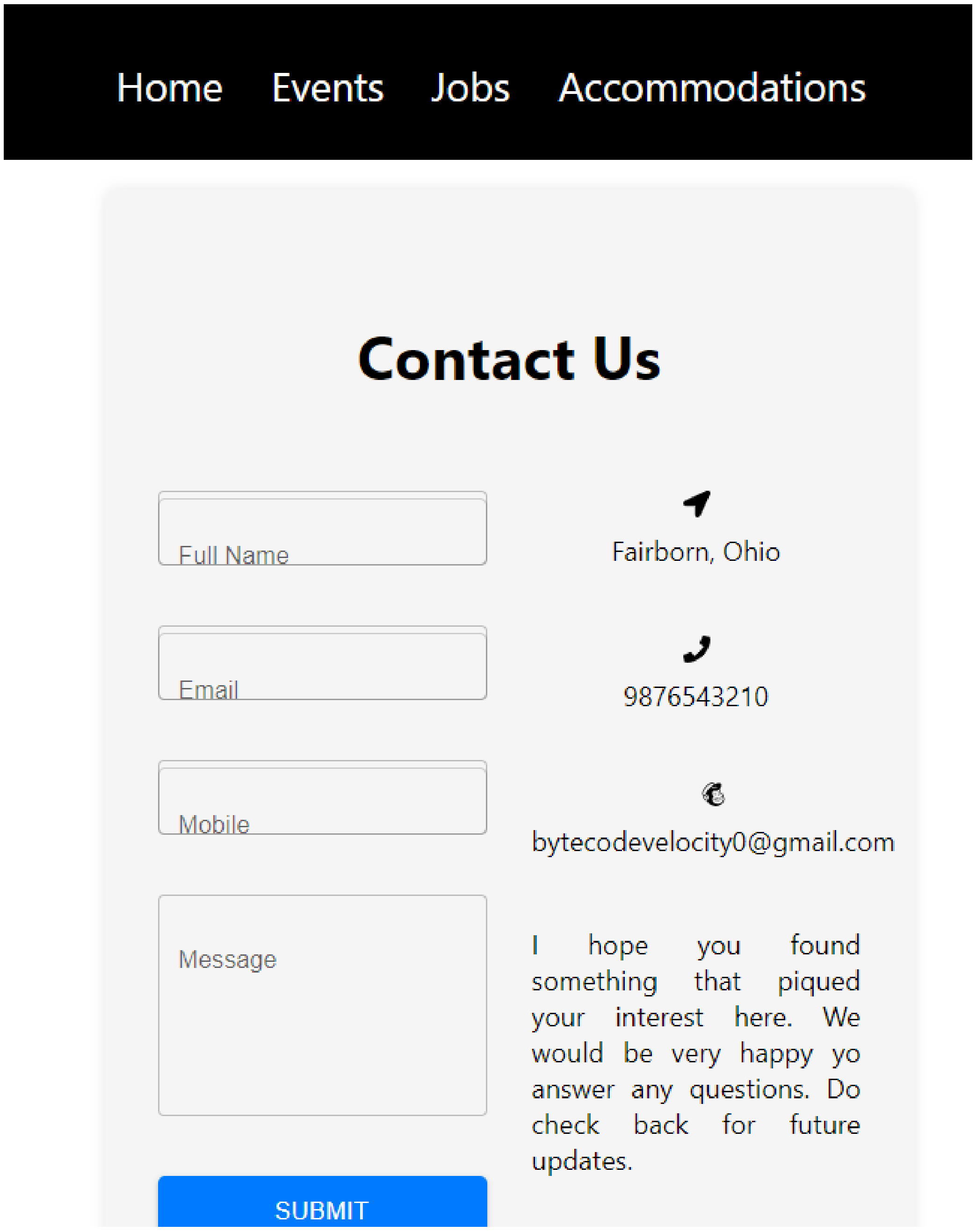
Job Description

- Lead the full software development lifecycle including requirements analysis design coding testing deployment and maintenance.
- Design and architect scalable reliable and maintainable software systems that meet performance and security requirements.

Job Detail Page



Apply Job Page



Contact Us

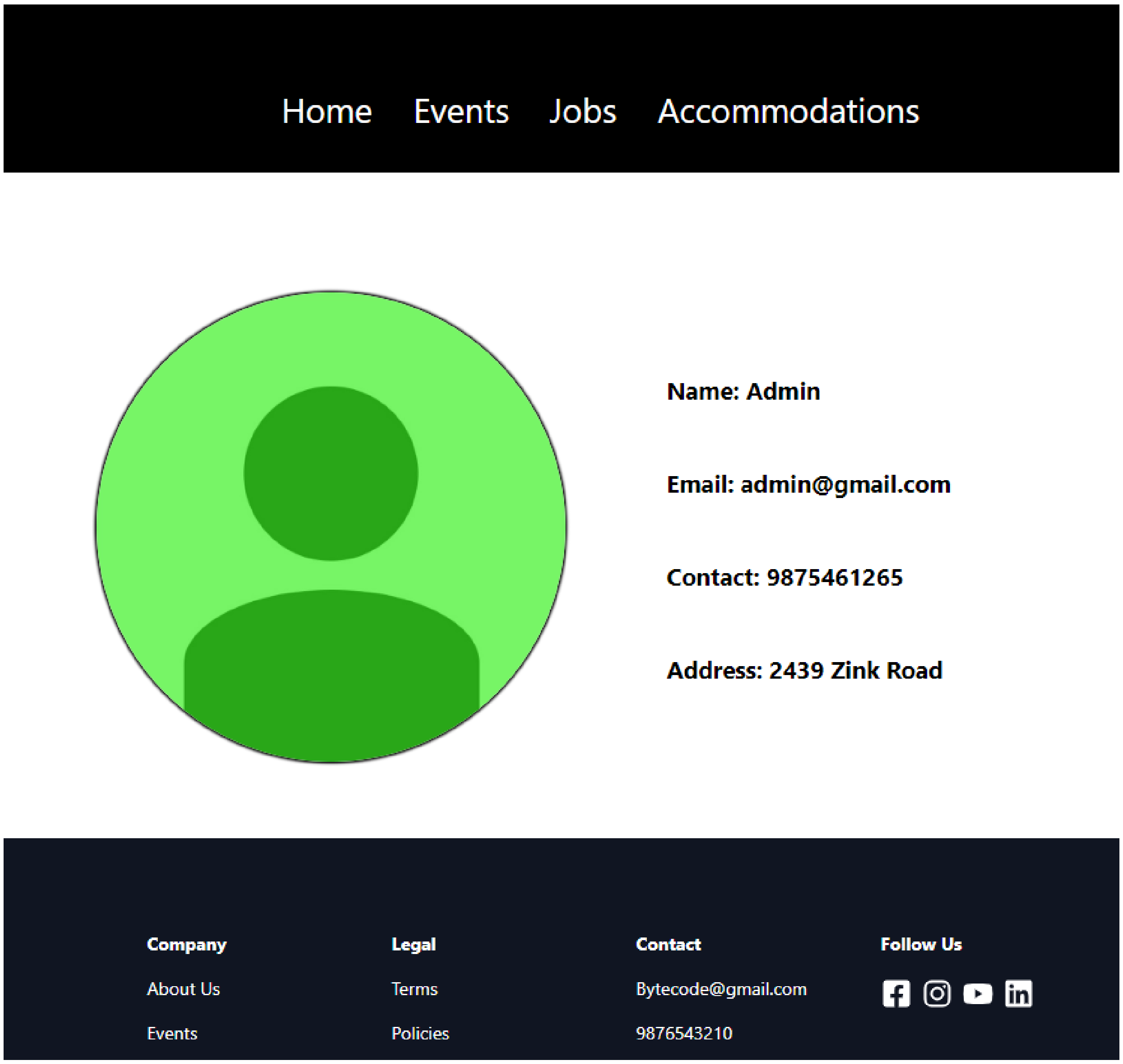
Fairborn, Ohio

9876543210

bytecodevelocity0@gmail.com

I hope you found something that piqued your interest here. We would be very happy yo answer any questions. Do check back for future updates.

Contact us page



Name: Admin

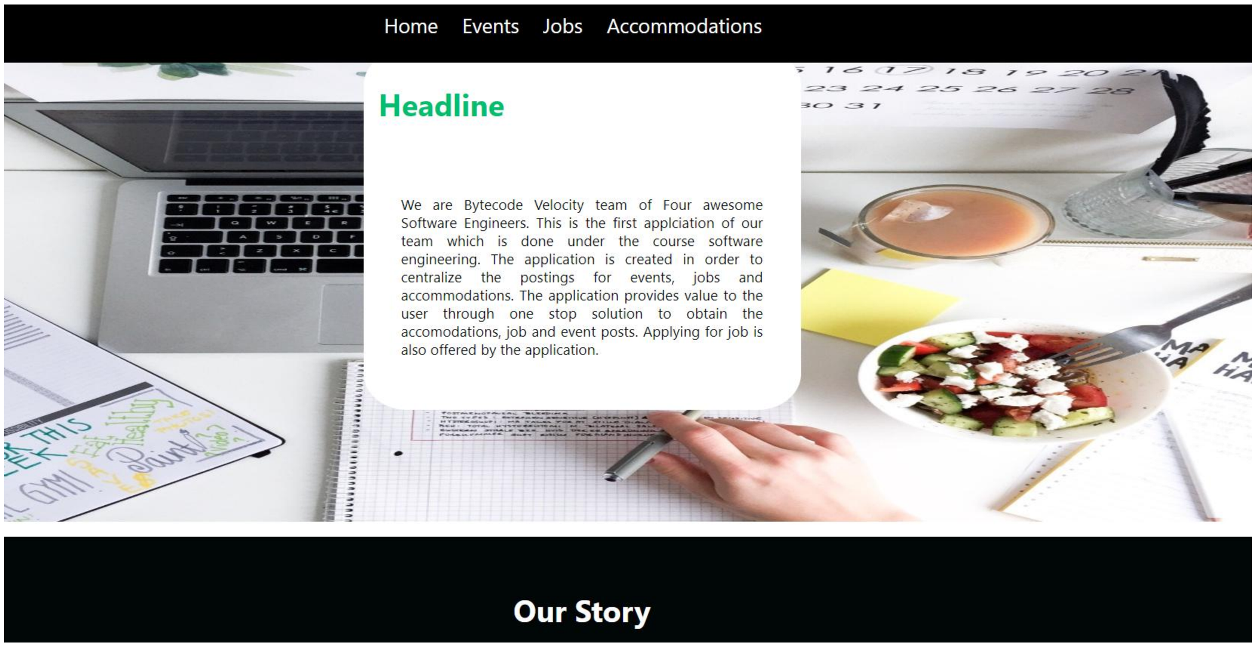
Email: admin@gmail.com

Contact: 9875461265

Address: 2439 Zink Road

Company	Legal	Contact	Follow Us
About Us	Terms	Bytecode@gmail.com	
Events	Policies	9876543210	

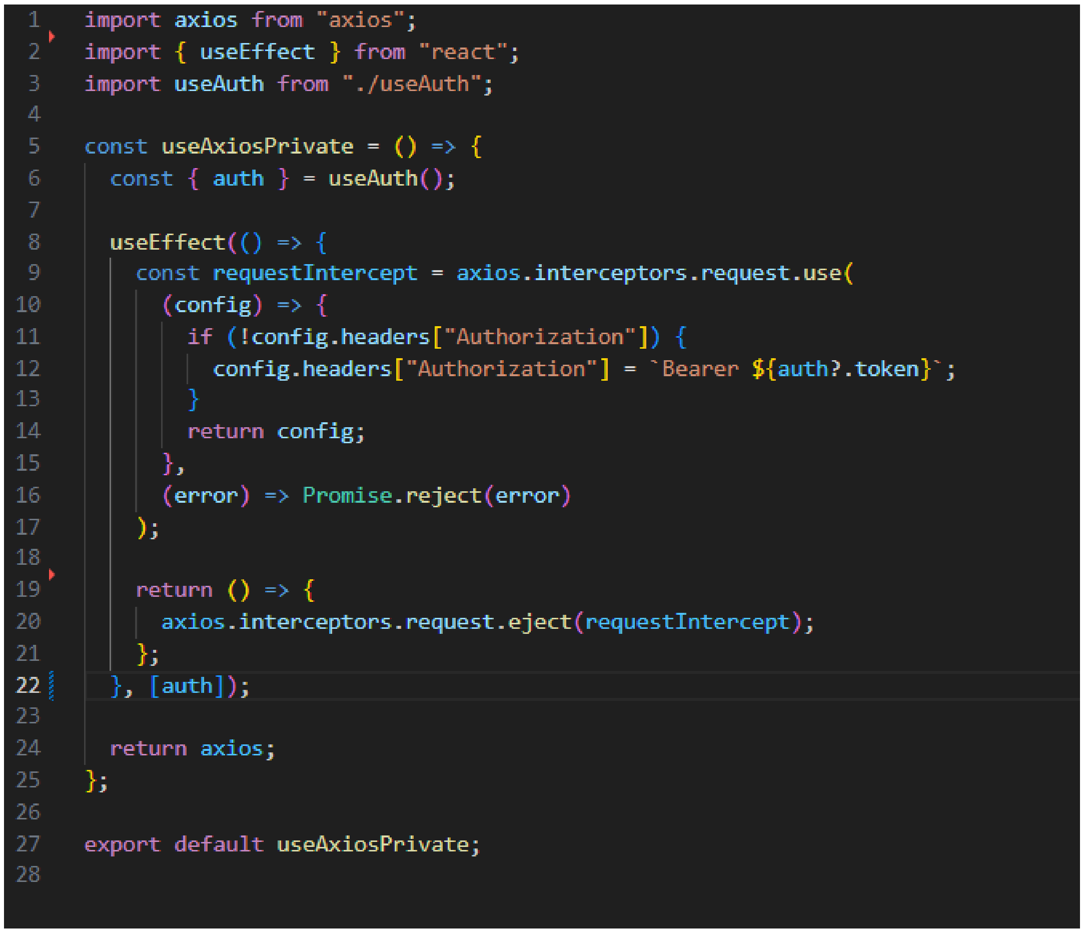
Profile View Page



We are Bytecode Velocity team of Four awesome Software Engineers. This is the first applciation of our team which is done under the course software engineering. The application is created in order to centralize the postings for events, jobs and accommodations. The application provides value to the user through one stop solution to obtain the accomodations, job and event posts. Applying for job is also offered by the application.

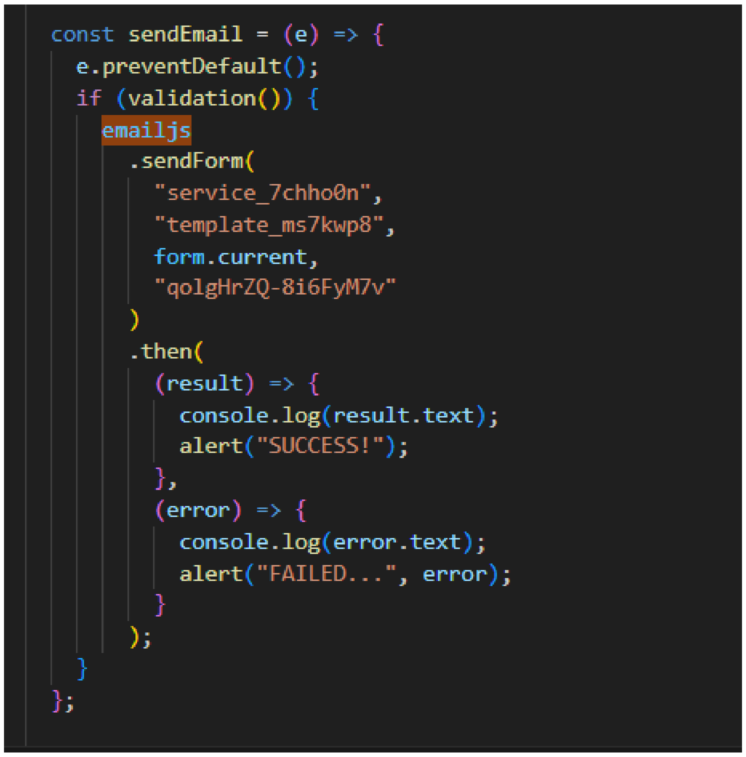
About-us page

Important Code Snapshots in frontend



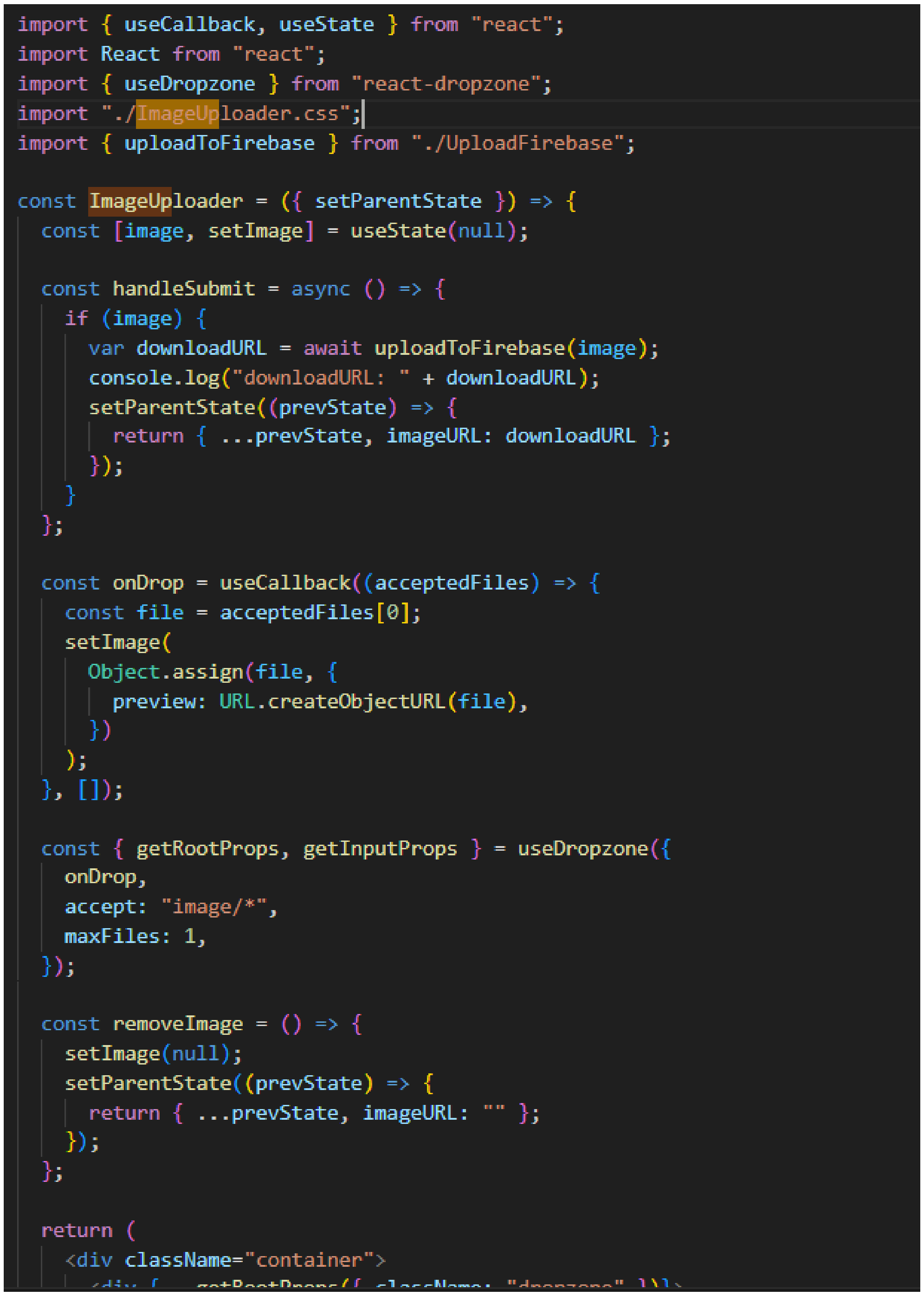
```
import axios from "axios";
import { useEffect } from "react";
import useAuth from "./useAuth";

const useAxiosPrivate = () => {
  const { auth } = useAuth();

  useEffect(() => {
    const requestIntercept = axios.interceptors.request.use(
      (config) => {
        if (!config.headers["Authorization"]) {
          config.headers["Authorization"] = `Bearer ${auth?.token}`;
        }
        return config;
      },
      (error) => Promise.reject(error)
    );

    return () => {
      axios.interceptors.request.eject(requestIntercept);
    };
  }, [auth]);

  return axios;
};

export default useAxiosPrivate;
```

Central axios implementation



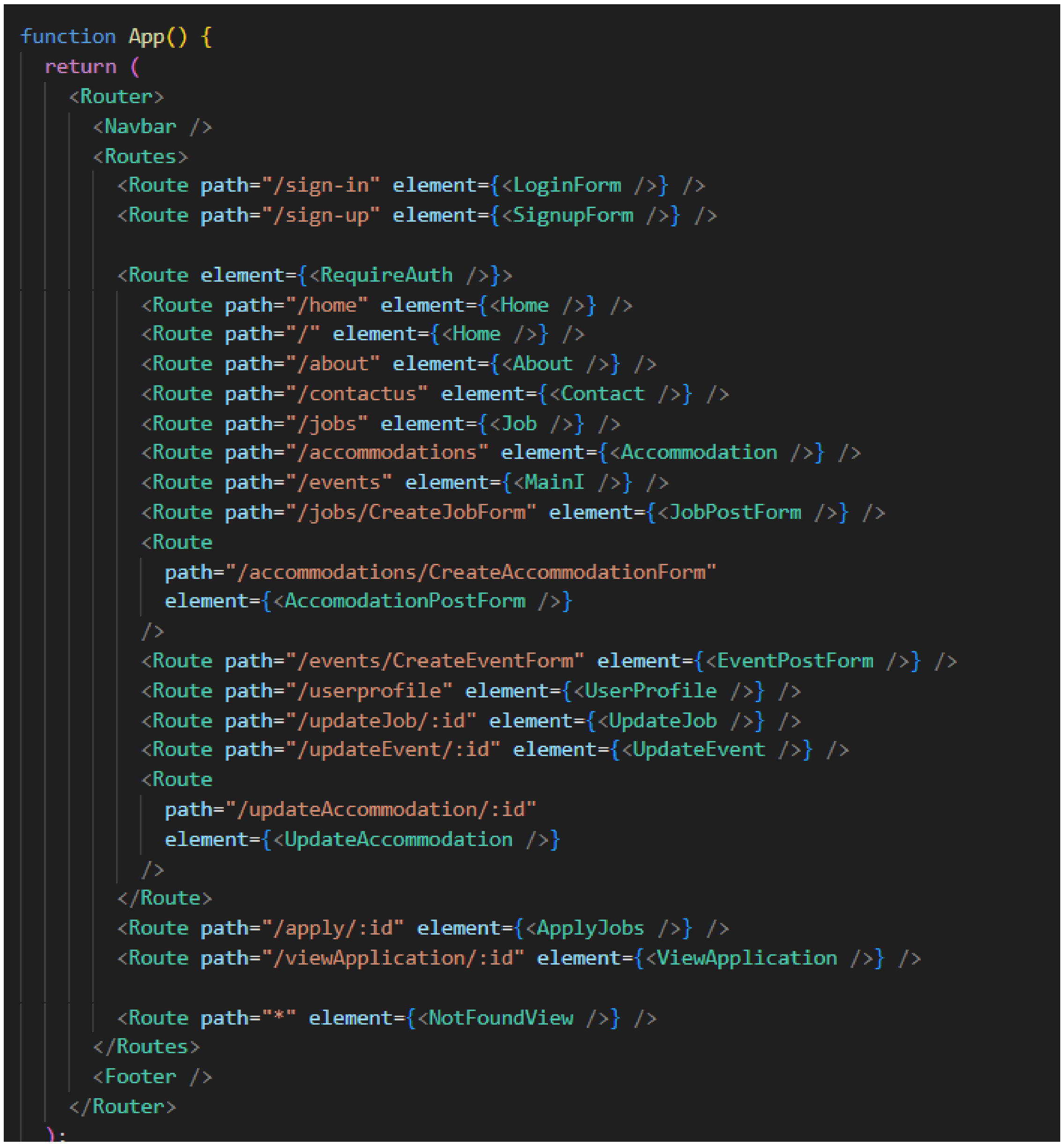
```
const sendEmail = (e) => {
  e.preventDefault();
  if (validation()) {
    emailjs
      .sendForm(
        "service_7chho0n",
        "template_ms7kwp8",
        form.current,
        "qolgHrZQ-8i6FyM7v"
      )
      .then(
        (result) => {
          console.log(result.text);
          alert("SUCCESS!");
        },
        (error) => {
          console.log(error.text);
          alert("FAILED...", error);
        }
      );
  }
};
```

Sending contact data through email



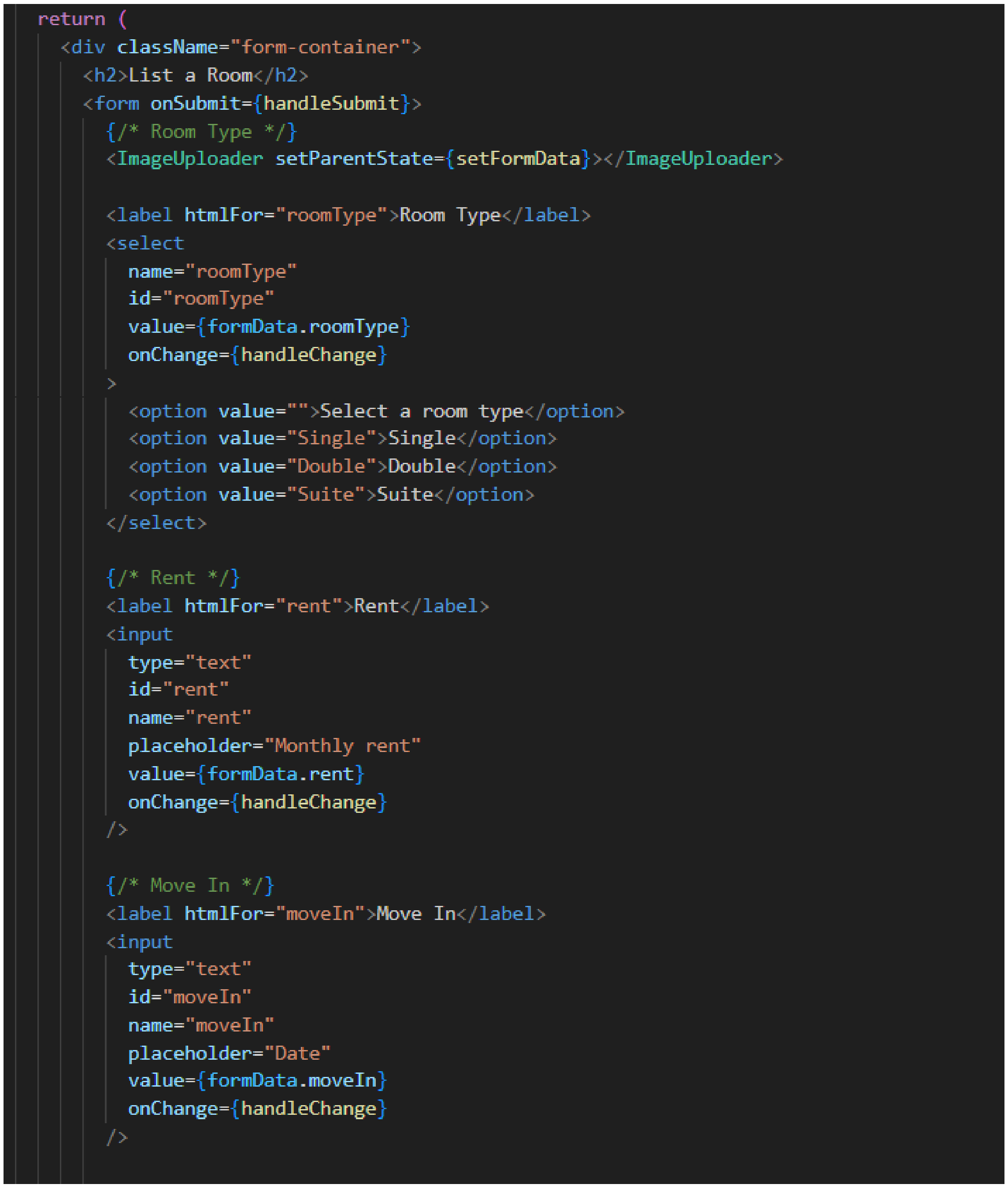
```
import { useCallback, useState } from "react";
import React from "react";
import { useDropzone } from "react-dropzone";
import "./ImageUploader.css";
import { uploadToFirebase } from "./UploadFirebase";

const ImageUploader = ({ setParentState }) => {
  const [image, setImage] = useState(null);

  const handleSubmit = async () => {
    if (image) {
      var downloadURL = await uploadToFirebase(image);
      console.log("downloadURL: " + downloadURL);
      setParentState((prevState) => {
        return { ...prevState, imageURL: downloadURL };
      });
    }
  };

  const onDrop = useCallback((acceptedFiles) => {
    const file = acceptedFiles[0];
    setImage(
      Object.assign(file, {
        preview: URL.createObjectURL(file),
      })
    );
  }, []);

  const { getRootProps, getInputProps } = useDropzone({
    onDrop,
    accept: "image/*",
    maxFiles: 1,
  });

  const removeImage = () => {
    setImage(null);
    setParentState((prevState) => {
      return { ...prevState, imageURL: "" };
    });
  };

  return (
    <div className="container">
      <div {...getRootProps({ className: "dropzone" })}>
```

Reusable and pluggable image upload component



```
function App() {
  return (
    <Router>
      <Navbar />
      <Routes>
        <Route path="/sign-in" element={<LoginForm />} />
        <Route path="/sign-up" element={<SignupForm />} />

        <Route element={<RequireAuth />}>
          <Route path="/home" element={<Home />} />
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contactus" element={<Contact />} />
          <Route path="/jobs" element={<Job />} />
          <Route path="/accommodations" element={<Accommodation />} />
          <Route path="/events" element={<MainI />} />
          <Route path="/jobs/CreateJobForm" element={<JobPostForm />} />
          <Route
            path="/accommodations/CreateAccommodationForm"
            element={<AccomodationPostForm />}
          />
          <Route path="/events/CreateEventForm" element={<EventPostForm />} />
          <Route path="/userprofile" element={<UserProfile />} />
          <Route path="/updateJob/:id" element={<UpdateJob />} />
          <Route path="/updateEvent/:id" element={<UpdateEvent />} />
          <Route
            path="/updateAccommodation/:id"
            element={<UpdateAccommodation />}
          />
        </Route>
        <Route path="/apply/:id" element={<ApplyJobs />} />
        <Route path="/viewApplication/:id" element={<ViewApplication />} />

        <Route path="*" element={<NotFoundView />} />
      </Routes>
      <Footer />
    </Router>
  );
```

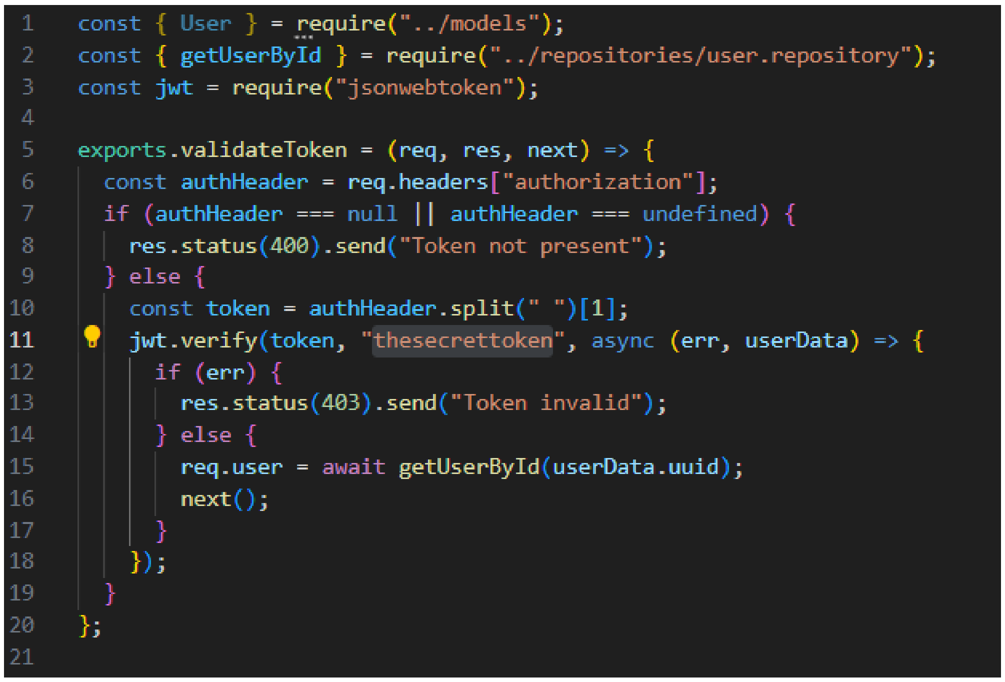
Routing implementation



```
  return (
    <div className="form-container">
      <h2>List a Room</h2>
      <form onSubmit={handleSubmit}>
        {/* Room Type */}
        <ImageUploader setParentState={setFormData}></ImageUploader>

        <label htmlFor="roomType">Room Type</label>
        <select
          name="roomType"
          id="roomType"
          value={formData.roomType}
          onChange={handleChange}
        >
          <option value="">Select a room type</option>
          <option value="Single">Single</option>
          <option value="Double">Double</option>
          <option value="Suite">Suite</option>
        </select>

        {/* Rent */}
        <label htmlFor="rent">Rent</label>
        <input
          type="text"
          id="rent"
          name="rent"
          placeholder="Monthly rent"
          value={formData.rent}
          onChange={handleChange}
        />

        {/* Move In */}
        <label htmlFor="moveIn">Move In</label>
        <input
          type="text"
          id="moveIn"
          name="moveIn"
          placeholder="Date"
          value={formData.moveIn}
          onChange={handleChange}
        />
```

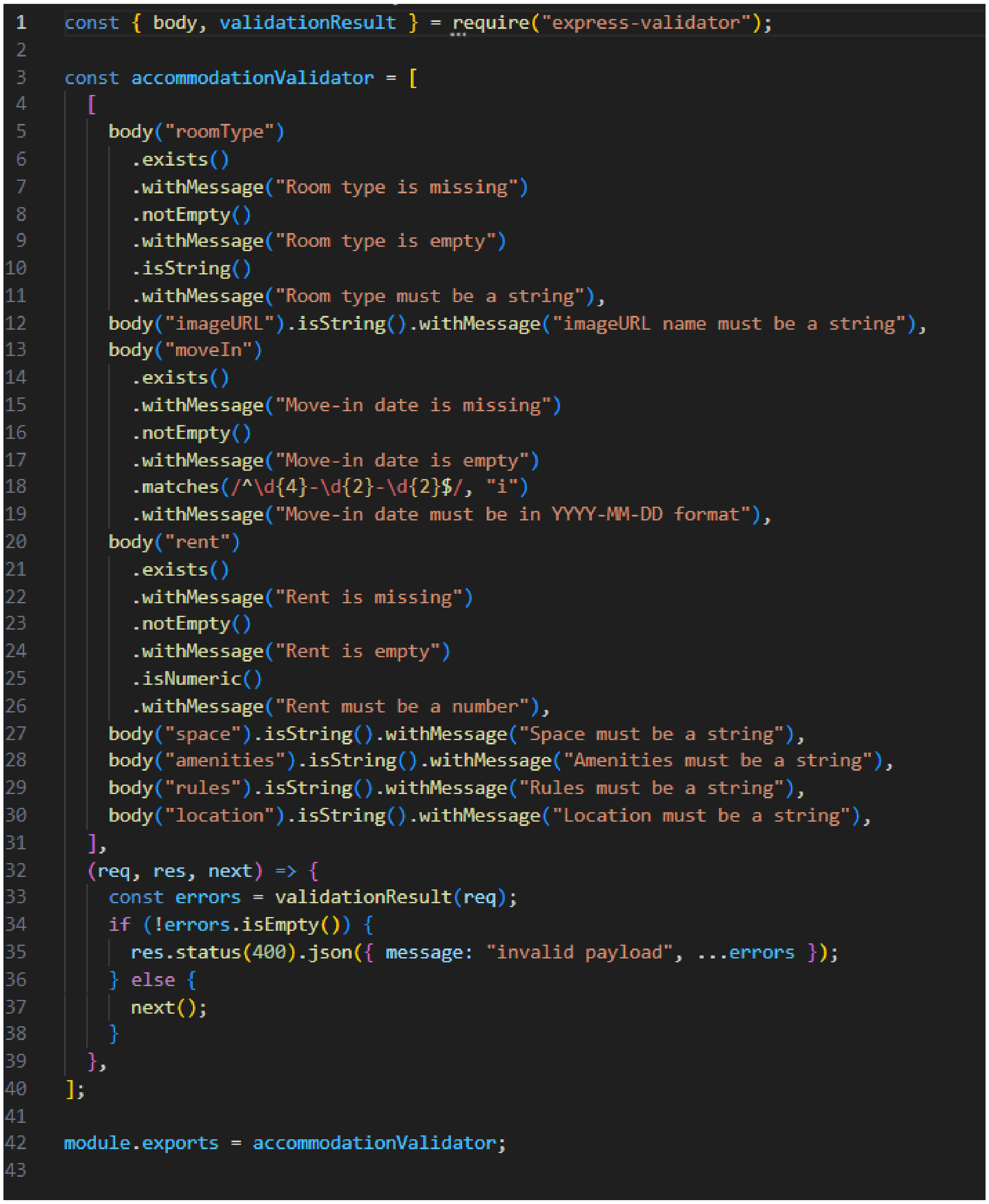
Create Accommodation Form

Important code Snapshots in Backend



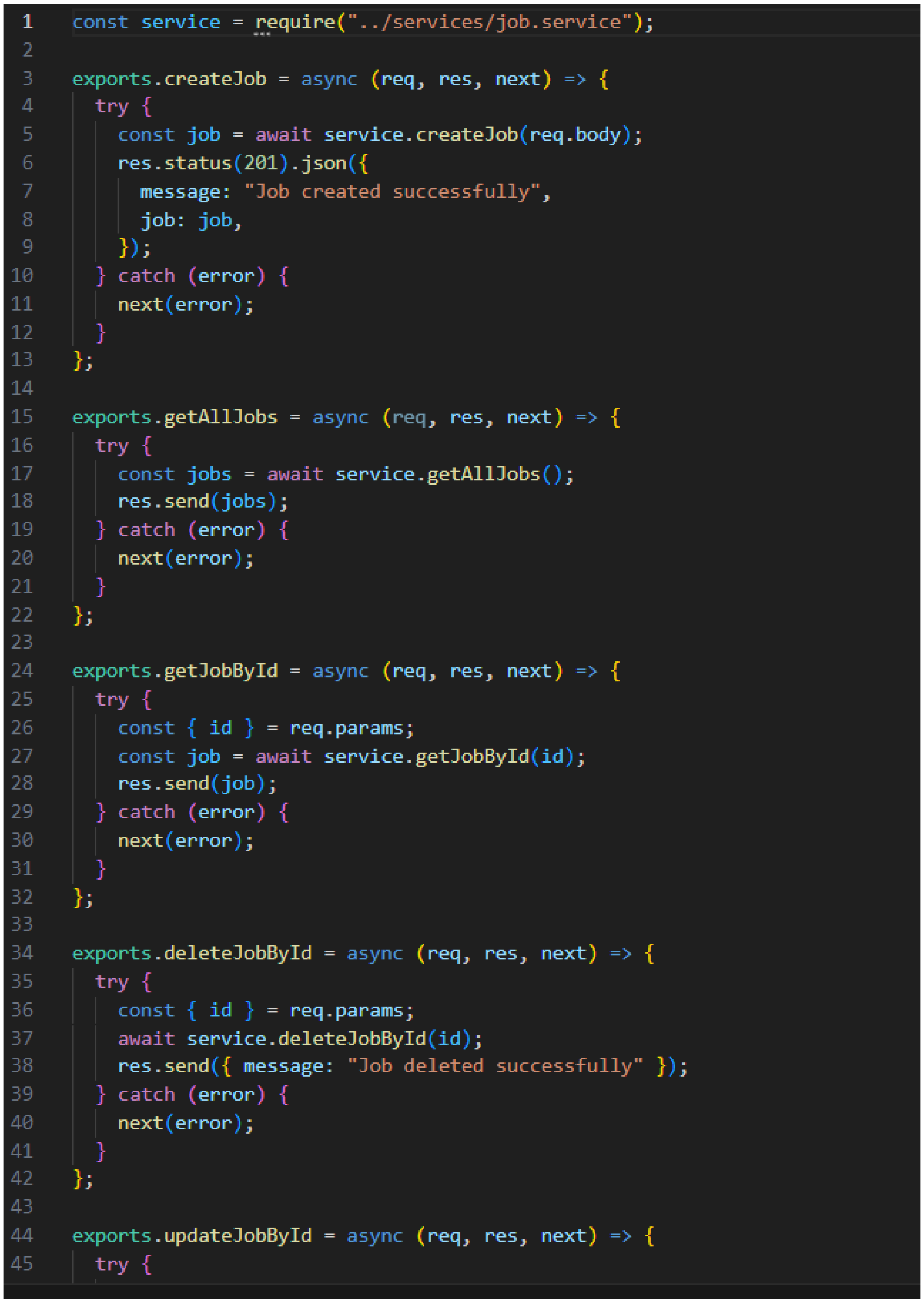
```javascript
const { User } = require("../models");
const { getUserById } = require("../repositories/user.repository");
const jwt = require("jsonwebtoken");

exports.validateToken = (req, res, next) => {
  const authHeader = req.headers["authorization"];
  if (authHeader === null || authHeader === undefined) {
    res.status(400).send("Token not present");
  } else {
    const token = authHeader.split(" ")[1];
    jwt.verify(token, "thesecrettoken", async (err, userData) => {
      if (err) {
        res.status(403).send("Token invalid");
      } else {
        req.user = await getUserById(userData.uuid);
        next();
      }
    });
  }
};
```

Token validator function

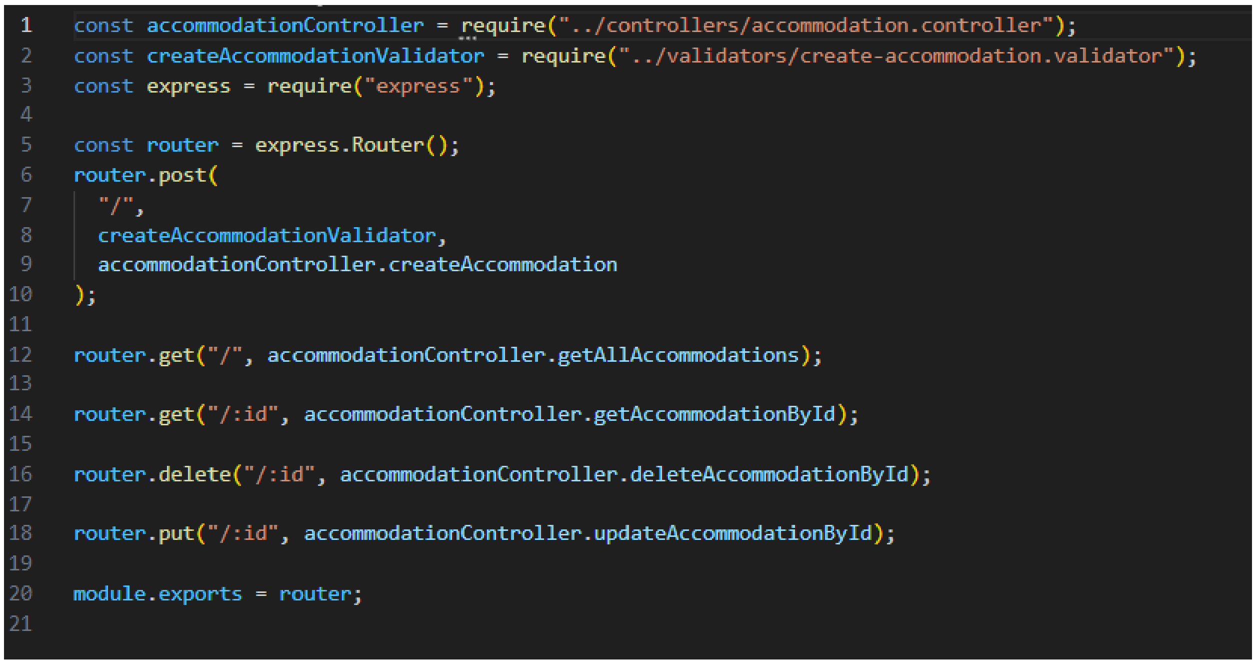


```javascript
const { body, validationResult } = require("express-validator");

const accommodationValidator = [
  [
    body("roomType")
      .exists()
      .withMessage("Room type is missing")
      .notEmpty()
      .withMessage("Room type is empty")
      .isString()
      .withMessage("Room type must be a string"),
    body("imageURL").isString().withMessage("imageURL name must be a string"),
    body("moveIn")
      .exists()
      .withMessage("Move-in date is missing")
      .notEmpty()
      .withMessage("Move-in date is empty")
      .matches(/^\d{4}-\d{2}-\d{2}$/, "i")
      .withMessage("Move-in date must be in YYYY-MM-DD format"),
    body("rent")
      .exists()
      .withMessage("Rent is missing")
      .notEmpty()
      .withMessage("Rent is empty")
      .isNumeric()
      .withMessage("Rent must be a number"),
    body("space").isString().withMessage("Space must be a string"),
    body("amenities").isString().withMessage("Amenities must be a string"),
    body("rules").isString().withMessage("Rules must be a string"),
    body("location").isString().withMessage("Location must be a string"),
  ],
  (req, res, next) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      res.status(400).json({ message: "invalid payload", ...errors });
    } else {
      next();
    }
  },
];

module.exports = accommodationValidator;
```

Data validator middleware

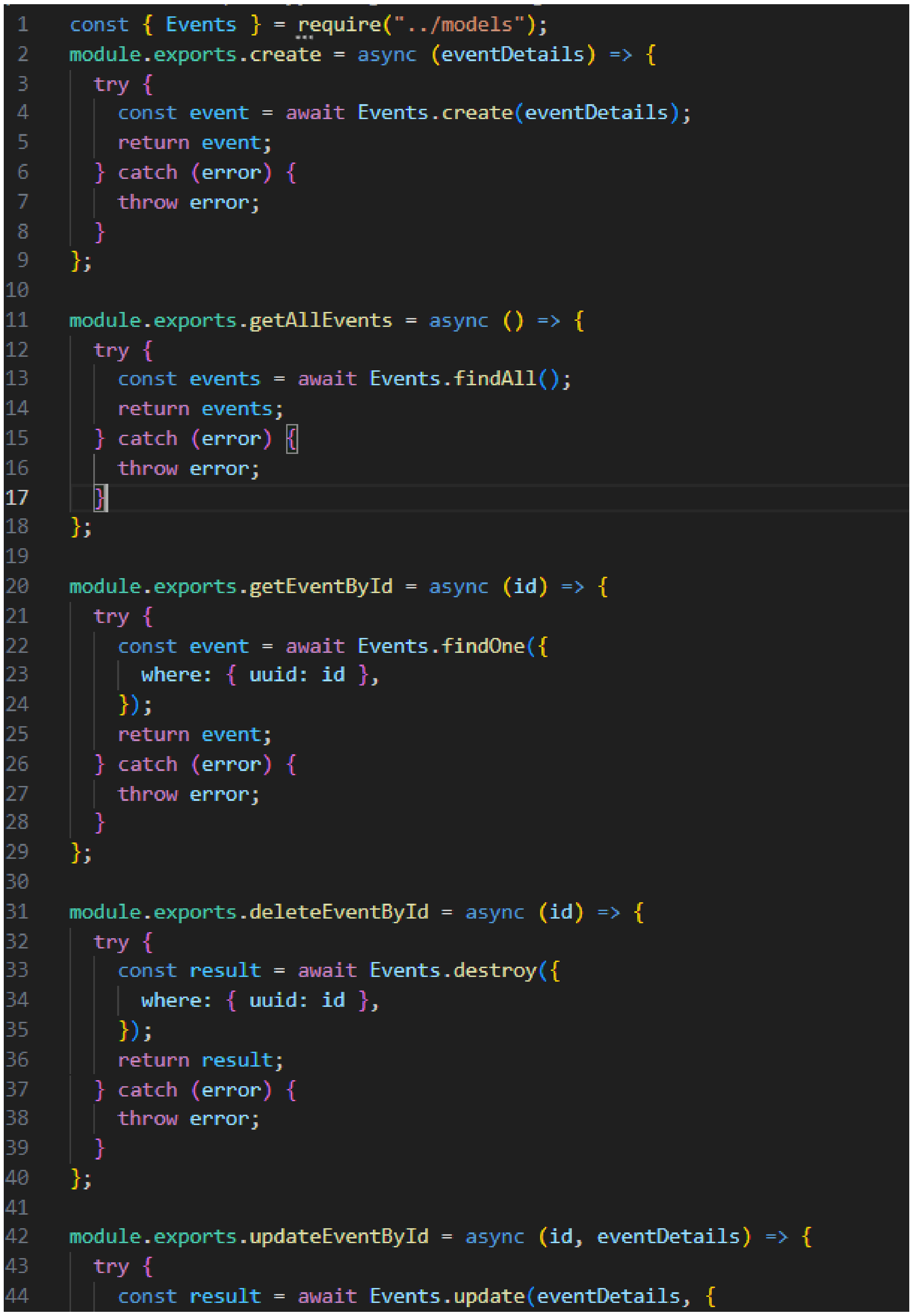

```javascript
const service = require("../services/job.service");

exports.createJob = async (req, res, next) => {
  try {
    const job = await service.createJob(req.body);
    res.status(201).json({
      message: "Job created successfully",
      job: job,
    });
  } catch (error) {
    next(error);
  }
};

exports.getAllJobs = async (req, res, next) => {
  try {
    const jobs = await service.getAllJobs();
    res.send(jobs);
  } catch (error) {
    next(error);
  }
};

exports.getJobById = async (req, res, next) => {
  try {
    const { id } = req.params;
    const job = await service.getJobById(id);
    res.send(job);
  } catch (error) {
    next(error);
  }
};

exports.deleteJobById = async (req, res, next) => {
  try {
    const { id } = req.params;
    await service.deleteJobById(id);
    res.send({ message: "Job deleted successfully" });
  } catch (error) {
    next(error);
  }
};

exports.updateJobById = async (req, res, next) => {
  try {
```

Model for job



```
const accommodationController = require("../controllers/accommodation.controller");
const createAccommodationValidator = require("../validators/create-accommodation.validator");
const express = require("express");

const router = express.Router();
router.post(
  "/",
  createAccommodationValidator,
  accommodationController.createAccommodation
);

router.get("/", accommodationController.getAllAccommodations);

router.get("/:id", accommodationController.getAccommodationById);

router.delete("/:id", accommodationController.deleteAccommodationById);

router.put("/:id", accommodationController.updateAccommodationById);

module.exports = router;
```

Accommodation Routes



```
const { Events } = require("../models");
module.exports.create = async (eventDetails) => {
  try {
    const event = await Events.create(eventDetails);
    return event;
  } catch (error) {
    throw error;
  }
};

module.exports.getAllEvents = async () => {
  try {
    const events = await Events.findAll();
    return events;
  } catch (error) {
    throw error;
  }
};

module.exports.getEventById = async (id) => {
  try {
    const event = await Events.findOne({
      where: { uuid: id },
    });
    return event;
  } catch (error) {
    throw error;
  }
};

module.exports.deleteEventById = async (id) => {
  try {
    const result = await Events.destroy({
      where: { uuid: id },
    });
    return result;
  } catch (error) {
    throw error;
  }
};

module.exports.updateEventById = async (id, eventDetails) => {
  try {
    const result = await Events.update(eventDetails, {
```

Event Repository